

A Low-Power 45nm 1Mb SRAM

Stevo Bailey, Kevin Linger, Roger Lorenzo, John Thompson

ECE 4332 – Fall 2011

University of Virginia

<sdb6t, kal5nx, rml6b, jrt6s>@virginia.edu

ABSTRACT

A low-power 1Mb SRAM was developed to fit the requirements set forth by Portable Instruments Company. The SRAM was primarily optimized at the bitcell and architecture levels. Emphasis was placed on developing a parameterized architecture model to aid in selecting the dimensions of the block that yielded the smallest energy consumption. We also focused on choosing optimal V_{DD} levels in normal and sleep modes, accounting for variation. Our circuit is reliable at all process corners and a range of temperatures. Our final metric was $4.552 \times 10^{-36} \text{ J}^2 \cdot \text{s} \cdot \text{mm}^2 \cdot \text{W}$.

1. INTRODUCTION

Portable Instruments Company (PICO) desired a low-power 1Mb SRAM with 32-bit words designed in the 45nm FreePDK technology for use in a microsensor node. The metric to be optimized was $(\text{Active Energy per Access})^2 \cdot \text{Delay} \cdot \text{Area} \cdot \text{Idle Power}$, showing how energy was the most important consideration. Additionally, the SRAM had to be robust across global process variations, temperatures, and supply voltages. To win the contract, we have successfully designed a low-power SRAM, using Cadence software, which meets the specifications.

2. SRAM OVERVIEW

An SRAM contains bitcells, which store one bit each, and periphery circuits for accessing the bitcells. The bitcells are partitioned into blocks. The energy and delay savings of partitioning offset the area overhead incurred [1]. Figure 1 shows the blocks and various periphery elements of our design.

Interleaving the columns within a block reduces the harmful effect of cosmic rays on one word. The bitlines access the data inside the bitcells when reading and writing. A precharge circuit charges the bitlines between cycles. When reading, a mux selects the desired word and sense amps detect a voltage difference between the bitline pairs. When writing, write amps pull one bitline of a pair low, and again a mux selects the word. The row decoder activates a word line, which turns on pass gates and connects the data in the bitcells to the bitlines.

When optimizing for low power, the bitcell was considered first. The next section gives the design choices made at the bitcell level. To simulate the memory, a model of the entire SRAM was designed; it is presented in section 4. Special features, including error correcting code (ECC) and sleep mode, appear in section 5. Section 6 expounds on the results of the project, and section 7 concludes this paper.

3. BITCELL

Since a bitcell is reproduced over a million times within a 1Mb SRAM, we spent ample time optimizing it. The standard 6T bitcell allows for a small overall area, relatively low leakage, and good stability. It contains two cross-coupled inverters and two pass gates for accessing the data. Simulations showed that minimum-sized transistors gave acceptable pull-up and cell ratios. The area of our bitcell was $0.816 \mu\text{m}^2$. We tested using high-threshold devices, but the delay overhead outweighed the energy savings. Reducing the supply voltage lowered energy but also

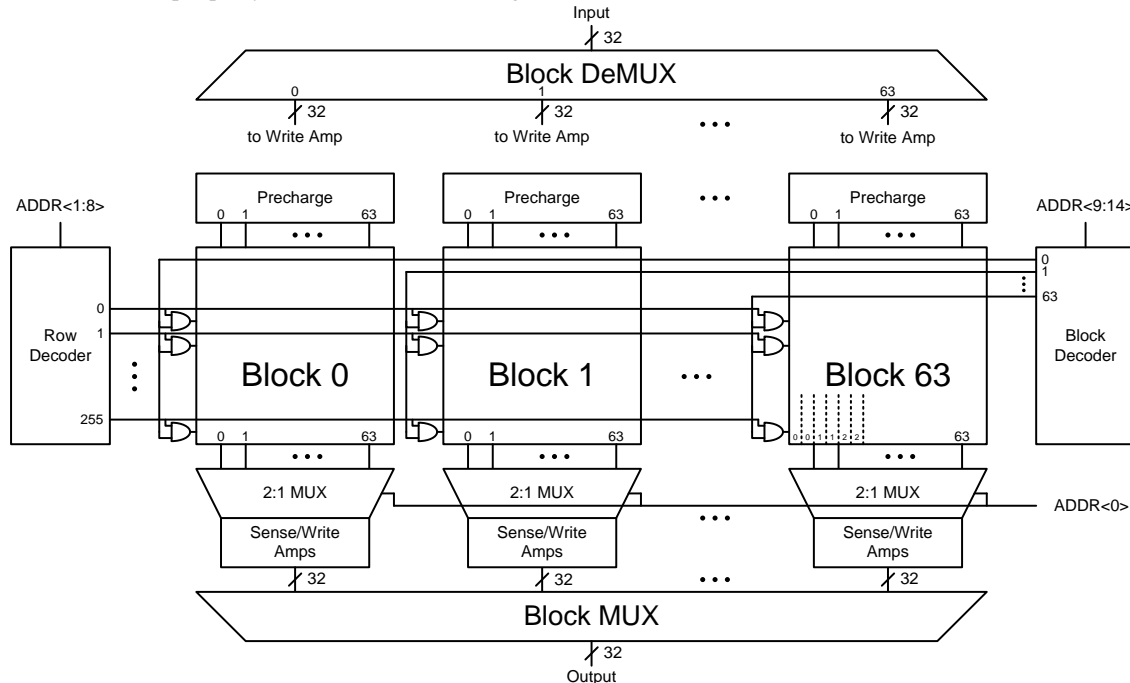


Figure 1. Global block diagram

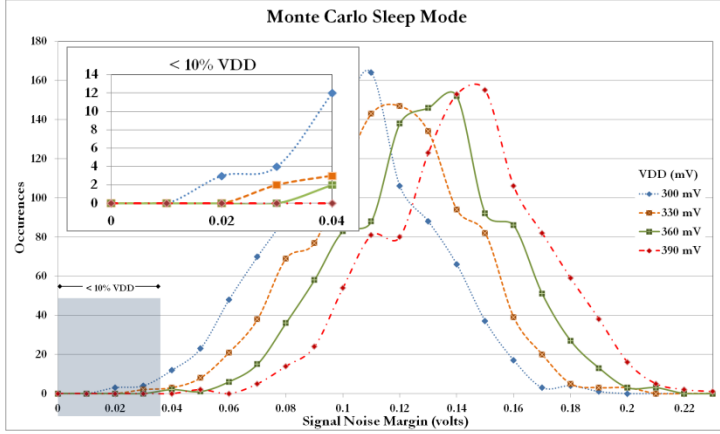


Figure 2. Monte Carlo analysis of sleep mode supply voltage and static noise margin; the inset shows the ideal sleep mode voltage is 390 mV

reduced the noise margin in the bitcell. Butterfly plots showed a worst-case noise margin of 78 mV for a supply voltage of 600 mV. As V_{DD} decreased, the signal to noise margin ratio worsened. Because of this and a previous group's choice, we chose a V_{DD} of 600 mV. Implementing a sleep mode further saves energy. To do this, we lowered V_{DD} such that the hold SNM is still large enough to retain stored data. Monte Carlo simulation results shown in Figure 2 verify that a 390 mV supply voltage ensures data retention. These bitcell optimizations gave our design a small area, low energy consumption, and sleep-mode stability despite local variation.

4. SRAM MODEL

In addition to optimizing the bitcells, we also optimized the architecture of the SRAM. We created a model which simulates reading and writing to a few bitcells for different numbers of rows, columns, and blocks. The model has two word lines, two bitline pairs, and all the necessary periphery circuits. Extra elements were added to approximate the loads on each node and the leakage from devices not on the simulated paths. To model a large number of elements, each having the same connections, the multiplicity (m) of transistors was used. A multiplicity of 2 represents two transistors with equal source, body, drain, and gate connections, but Spectre treats the transistor as one with upsized parameters. This reduces computation during simulation. By creating this parameterized model, we encapsulated the features of the SRAM that change with block dimensions, and chose the optimal dimensions to meet our metric.

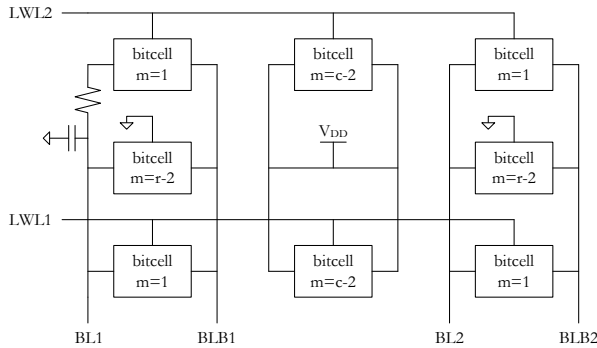


Figure 3. Model for one block

4.1 Blocks

Figure 3 shows the block model. It contains four accessible bitcells of multiplicity 1. Four extra bitcells approximate the load on the word lines and bitlines and the leakage of the other bitcells. Their multiplicities depend on the number of rows (r) and columns (c). An additional block model was used to model the leakage in all the other blocks. Since the bitlines are long, an RC load was added to model the parasitics of the wires. The resistance and capacitance values were obtained from the NCSU wiki [3] and parasitic extraction, respectively.

4.2 Precharge

To precharge the bitlines, two PMOSs connect them to V_{DD} during the precharge phase of the cycle. A third PMOS couples bitline pairs during precharge since a difference in leakage could create a significant voltage differential across the bitline pairs before the read cycle begins. Extra precharge blocks modeled the load on the precharge signal. The charging PMOSs were upsized to speed up charging of the bitlines. Because the precharge signal sees a large number of upsized gates, the signal was buffered, creating a global and local precharge signals, before entering each block.

4.3 Decoders

There are three decoders: row, word, and block. To reduce the number of transistors and the delay of the decoders, we implemented predecoders [2]. The decoders were modeled as AND gates in series representing the predecoding and final decoding stages. Additional off-path AND gates were used as loads.

4.4 Word Lines

To reduce the delay of the word line signals, the word lines are made hierarchical. Global word lines (GWL) come from the row decoder, and a block decoder signal selects which local word line (LWL) receives the GWL signals. An extra AND gate simulates the load on the GWL from the other blocks. As in the block model, RC loads were added to the GWLs and LWLs. The word line model can be seen in Figure 4.

4.5 Read/Write Muxes

Two muxes allow each bitline pair to connect to either a sense amp or a write amp. Their outputs are shared with the outputs of muxes from other columns. Since writing requires completely discharging one bitline in a pair, the write mux was upsized to allow more current through it. This reduced the write delay.

4.6 Sense Amps

Two sense amp topologies, taken from the class lecture and seen in [4], were considered. The optimal design had the smallest

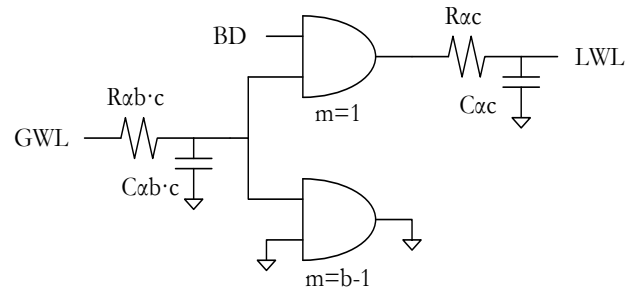


Figure 4. Model for the word lines

delay, since the energy and area differences were negligible compared to the energy and area of all the bitcells. Sense amp 2 had a smaller delay, so it was used. In the model, extra sense amps were included to approximate the leakage from sense amps not used during that cycle.

4.7 Write Amps

To amplify the discharge of one bitline in a pair, NMOSs are used. Inverters buffer the output of the block demux from the amplifiers. Off-path NMOS amplifiers are included to estimate the load. The write amps were upsized to speed up writing, making read the critical path of our design.

4.8 Block Mux/Demux

Since the block mux and demux are transmission gate trees, we modeled them by one path of a transmission gate tree with off-path loads. Because the tree simplifies down to pass gates in series, buffers were inserted at intermediate nodes to attenuate noise that flows through the mux.

5. SPECIAL FEATURES

Special features extend the functionality of our SRAM. Since error correcting code (ECC) is not in the project description and is separate from the SRAM, its overhead is not included in the metric.

5.1 Optimal VDD

In addition to using static noise margin and Monte Carlo analysis to determine the lowest possible V_{DD} during active and sleep modes, we also determined the effect of supply voltage on our metric. Although delay decreased, total energy increased for higher V_{DD} values, resulting in an exponential growth of our metric with respect to an increasing supply voltage, as seen in Figure 5.

5.2 Error Correcting Code

We implemented a single error correction, double error detection (SECDED) (39, 32) Hsiao code [5]. The Hsiao code gives a more regular architecture over a Hamming code implementation because its columns are minimally odd-weighted. Simulations proved the functionality of the circuit. Implementing the code resulted in a worst-case delay overhead of 0.36 ns and energy overhead of 53.4 fJ per read. Because our optimal block size contains only 2 words per row (see section 6), interleaving just alternates the bits in the words. A cosmic ray burst would likely

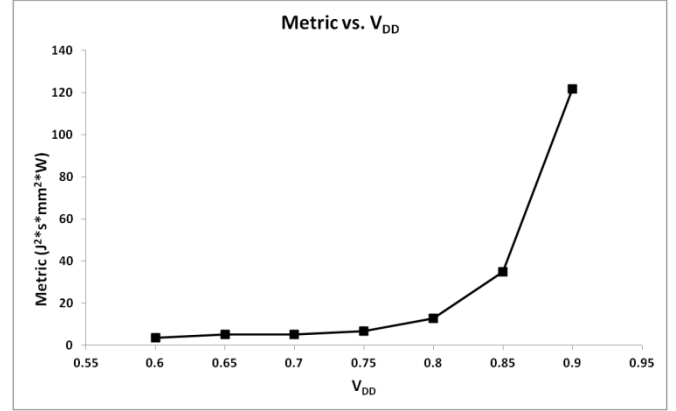


Figure 5. The total metric rises exponentially for an increasing supply voltage

disrupt more than one bit in a word. This means our ECC would not be useful. While this is a limitation to our design, it can easily be remedied by increasing the number of correctable bits or increasing the number of words in each row.

5.3 Sleep Mode

To take advantage of a larger noise margin during a hold, our SRAM lowers V_{DD} during periods of inactivity. Monte Carlo simulations gave a data retention voltage of 390 mV. Having this sleep mode reduces the idle power by more than two thirds.

6. RESULTS

We simulated the model and swept the number of rows, columns, and blocks. For each case, we found the energy and delay of the SRAM. Figure 6 shows the results obtained. Figure 7 shows an enlarged version of the five optimal data points after additional considerations were decoders were added to the model. The optimal dimensions were 256 rows, 64 columns, and 64 blocks. While the dimensions we chose did not have the lowest energy, the E^2D products of the three best design points were comparable. Our model did not account for changes in area. An increase from 64 blocks to 128 blocks incurs significant area overhead due to a large increase in the size of the block mux and demux. Because of this, we chose 64 blocks. The metric is given by

$$\text{Metric} = (\text{Active Energy per Access})^2 \cdot \text{Delay} \cdot \text{Area} \cdot \text{Idle Power}.$$

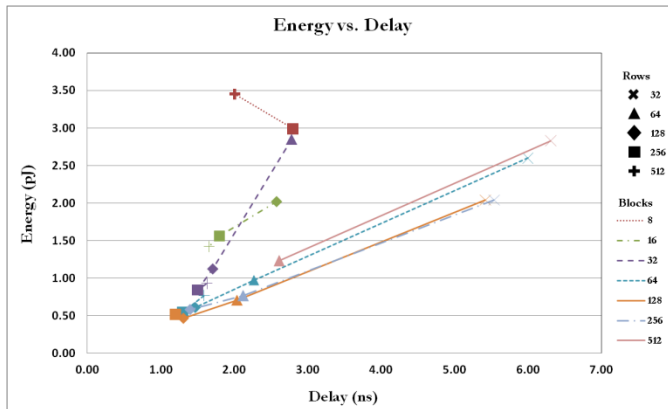


Figure 7. Energy v. Delay curve after sweeping the model parameters

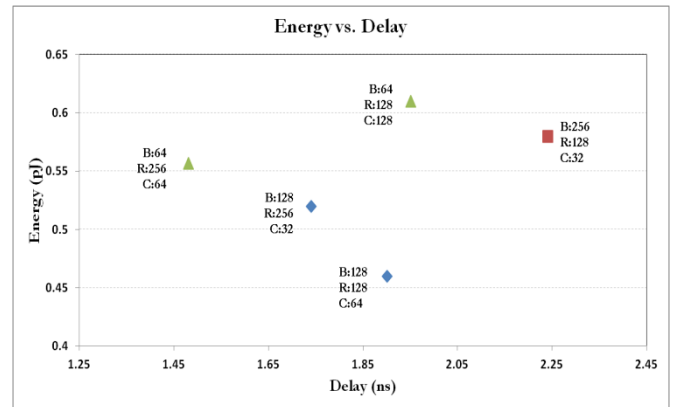


Figure 6. Energy v. Delay of the optimal points

6.1 Energy

PICo defined the total energy as the average energy per 32-bit access given an average of 5 reads for each write operation. To calculate this metric, we separated the current drawn by the bitlines (since we only simulated one bit) from the current drawn by everything else. We multiplied the bitline current by 64 during a read since all bitcells in a row are read. In a write, 32 bits are written to, and the other 32 bits in the row perform a dummy read. Accounting for this, we averaged the current and multiplied by V_{DD} and the time, then divided by the six operations. Our total energy was 2.69 pJ.

6.2 Delay

Delay is defined as the worst case delay for either a read or a write. A read and write includes precharging the bitlines after the operation. However, since precharging is relatively quick, the duty cycle of the clock was altered to prevent wasted time during precharge. For a read, the voltage differential across the bitline pairs required to prevent the sense amp from misreading due to local process variations was 200 mV (see [6] pp. 48-49). Reading took the longest. The SS process corner required a longer clock period than TT to generate the 200 mV bitline differential. This is reflected in Figure 8, which shows a TT timing diagram for a read and a write. Our worst-case read delay was 2.9 ns.

6.3 Area

The area was prescribed as a rectangular box that completely encompassed the SRAM layout. Most of the area was consumed by the bitcell blocks and the global word line wiring. Periphery circuits accounted for only 19.1% of the total area. Our final area was 1.02 mm².

6.4 Idle Power

Idle power was determined by the average current during an idle period times the supply voltage. Our SRAM has an idle power of 208 μ W.

6.5 Final Metric

The final metric does not include overhead from the ECC, and it does not account for the power savings in sleep mode. We kept the supply voltage at 600 mV. The value of our final metric was $4.539 \times 10^{-36} \text{ J}^2 \cdot \text{s} \cdot \text{mm}^2 \cdot \text{W}$.

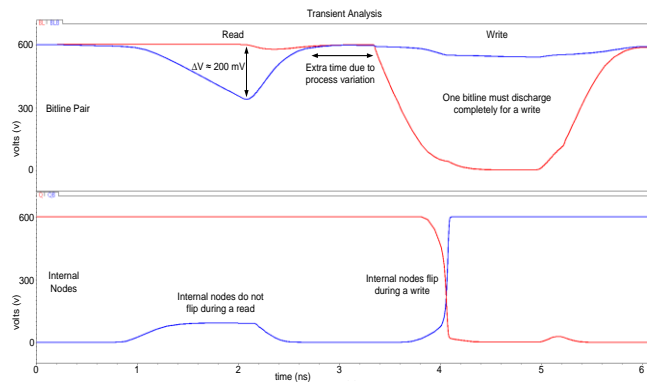


Figure 8. Timing diagram of one read and one write

Table 1. Final metric calculation

Metrics	
Total Energy	2.69 pJ
Total Delay	2.9 ns
Total Area	1.02 mm ²
Idle Power	208 μ W
Final Metric	$4.539 \times 10^{-36} \text{ J}^2 \cdot \text{s} \cdot \text{mm}^2 \cdot \text{W}$

7. CONCLUSION

To optimize energy, we lowered the supply voltage to 600 mV. We included a sleep mode, which decreased the power consumed when holding by more than two thirds. We tested the energy savings of dual-VT and found them to be negligible. Finally, we optimized the structure of our SRAM to approach the pareto optimal curve. The energy optimization schemes resulted in a total energy per cycle of about 1.5 pJ. Last year's two groups had total energies of 10 fJ (low power) and 2.5 nJ (high speed). Considering that charging one of our bitlines consumes more energy than the low power group's entire SRAM, we believe that their energy consumption is unreasonable. While we optimized our circuit for energy, its delay is still comparable to the high speed group.

Limitations hinder the effectiveness of our model. Only the bitcells were verified to work under local process variations. Each component in the model estimates capacitive loads and leakage consumed; the accuracy of these models could be improved by limiting the amount of modeling and testing a larger portion of the SRAM. Future work should also include better error correcting by increasing the number of words per row or improving the ECC to correct for more errors.

8. ACKNOWLEDGMENTS

The authors would like to thank the University of Virginia for providing the technological resources used for our research, Professor Benton Calhoun for assisting us along the way, and the teams from previous years for giving advice on and insight into designing our SRAM.

9. REFERENCES

- [1] Amrutur, B. S. and Horowitz, M. A. Speed and power scaling of SRAM's, *IEEE Trans. on Solid-State Circuits*, 35, 2 (Feb. 2000), 175-185.
- [2] Rabaey, J. M., Chandrakasan, A., and Nikolić, B.. Digital integrated circuits: A design perspective, 2nd ed., Upper Saddle River: Pearson, 2003, 674.
- [3] http://www.eda.ncsu.edu/wiki/FreePDK45: Metal_Layers
- [4] Jacob, B., Ng, S. W., & Wang, D. T. Memory systems: cache, DRAM, disk, Burlington: Morgan Kaufmann, 2007, 282.
- [5] Hsiao, M. Y. A class of optimal minimum odd-weight column SEC-DED codes, *IBM JR & D*, 14, (July 1970), 395-403.
- [6] Verma, N. *Ultra-Low-Power SRAM Design In High Variability Advanced CMOS*, Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, 2009.